Design and evaluation of a learning environment to effectively provide network security skills

Abstract

Information system security and network security are topics of increasing importance in the information society. They are also topics where the adequate education of professionals requires the use of specific laboratory environments where the practical aspects of the discipline may be addressed. However, most approaches currently used are excessively static and lack the flexibility that the education requirements of security professionals demand. In this paper we present NEMESIS, a scenario generation framework for education on system and network security, which is based on virtualization technologies and has been designed to be open, distributed, modular, scalable and flexible. Finally, an example scenario is described and some results validating the benefits of its use in undergraduate computer security courses are shown.

Keywords: Architectures for educational technology systems, Cooperative/collaborative learning, Distributed learning environments, Interactive learning environments, Simulations, Teaching/learning strategies

1. Introduction

Information systems and networks play a paramount role in the contemporary societies. The degree of integration of these systems and networks in our daily lives is such that it is extremely difficult, if not impossible, to imagine our societies without them. Information systems are a key component in most government and enterprise structures. This dependency of the social infrastructures on the information systems supporting them makes societies as vulnerable as their underlying information systems are, which makes system and network protection a critical aspect (Shaw et al., 2009).

To protect these systems and networks, there are two main categories of threats to take into account: physical threats and logical threats. If we analyze data from the last years, we can observe there has been a continuous increase in the number and severity of logical attacks to critical infrastructures (Cardenas et al., 2009). This is specially true in the case of electronic attacks. For this reason, it is necessary to devote effective and efficient efforts to protect our infrastructures against these threats, and one of the critical requirements in order to achieve this is to be able to provide high quality education on information system and network security. This is not a new requirement, however. Since the 90s, different experts have recommended to strengthen the security-related subjects in the curriculums of network and computer science disciplines, pointing out that education on these matters is insufficient, and do not cover adequately the real needs (Yang et al., 2004).

Though there are different approaches to introduce system and network security in computer science courses, most of them distinguish two fundamental blocks: information security, where

the main emphasis is put on the use of cryptographic techniques, and system security (Higgins, 1989). Specially in the latter case, one of the key requirements in order to provide a complete high-quality education is to have the adequate equipment. Due to the practical nature of the discipline, it is very difficult to address effective teaching on these topics without a lab environment where the practical aspects of the discipline may be studied (Lee et al., 2011). However, even where there are resources to build this kind of labs, a common drawback is the difficulty to provide scenarios similar to real environments, which make students face challenges which are closer to those they would find in the real professional experience. In addition, due to the nature of the topics addressed in these subjects, system administrators are reluctant to the deployment of security labs in campus networks, unless they are isolated from the rest of the network. Therefore, most system security labs we can find are very limited, while most teachers would agree that the ideal case would be to provide the students with practical environments where they can interact with realistic systems using state-of-the-art technologies.

The design of realistic security labs is thus a challenge for the teachers of this kind of courses. In this line, our research aims to design security labs with realistic scenarios without negatively impacting the campus network performance, and in a modular and extensible way which allows the labs to be adapted to the rapid evolution of the discipline. This paper contributes to this goal in the following ways:

- We analyze the problem of teaching network and system security using scenarios, and we review the main existing approaches to teach this field (Section 3).
- We establish a set of requirements for a solution to generate and deploy realistic security scenarios (Section 4.1).
- We propose a hierarchical, modular model to define scenarios for network and system security in a flexible and extensible way (Section 4.2).
- We design an architecture based on virtual machines distributed among different physical machines, which allows to deploy complex scenarios in a scalable manner (Section 4.3).
- We provide a schema for scenario description based on templates and XML file descriptions, which allows for high expressiveness and flexibility in scenario definition (Section 4.4).
- We provide illustrative examples on the practical use of the proposed framework to teach system security (Section 5).

To validate the proposed framework, we provide a use case on its use as a support tool for network security education (Section 6), along with the discussion of the results obtained. Finally, the last section summarizes our contributions and sheds light on some future research lines.

2. Teaching Network and System Security

There are several traditional approaches to teach Network and System Security. Most courses we can find in universities differ both in the variety of topics covered and in the methodology used to address them. Regarding the contents of the courses, we can find basically three types: cryptography courses, system security courses and survey courses (Bishop, 1993). Cryptography courses are focused on the analysis, design and use of cryptographic mechanisms, and they

usually rely heavily on mathematical foundations of cryptography, such as information theory, number theory and statistics. System security courses tend to draw apart from the foundations of cryptography (cryptography is seen here just as one more tool in the security professional toolbox) and focus on the design of secure information systems, usually covering the concepts of security threats, vulnerabilities and mechanisms to address them. Depending on the nature of the course, and usually of the context within the degree, there may be more emphasis in low-level technical details (such as vulnerability exploitation) or in high-level design principles (such as minimum privilege, segregation of duties or risk analysis). Finally, survey courses intend to give a general view of all the security-relevant topics, and thus cover both cryptography and network security in a more shallow way.

Regardless of the nature of the course from the point of view of its contents, there are different methodologies that have been traditionally used to teach information and network security (Yurcik & Doss, 2001; Sharma & Sefchek, 2007; Khambari et al., 2009). The traditional lecture is probably the dominant approach we can find. Although this approach is probably justified in cryptography courses due to the need to understand the mathematical foundations of the discipline, it has the risk of becoming too descriptive and requiring so little involvement by the students that they may become too passive, and thus the learning process may not work adequately. Of course, techniques for making the lecture class more participative and appealing have been used, like providing real-life examples or in-class exercises. However, such an approach has the drawback of not incentivizing creative and lateral thinking, which are crucial elements in information system security.

Given the vastness of the security related topics, a growing tendency is to complement lectures with supplementary materials such as tutorials. There is a huge amount of freely available material on information security, which may be given to students as further readings and references in case they want to expand their understanding of a particular topic. This is usually of great help to keep the students motivated, since a broad offer of supplementary material increases the likelihood of finding a topic of particular interest to the student. However, most of this material is theoretical in nature, and even the practical material is very linearized (step-by-step howtos, for instance), so there is still very little room for the students to develop discipline-related free thinking.

Finally, most current security courses in higher education complement the lecture approach with lab assignments. Lab assignments intend to put the student in contact with some of the topics in the discipline in a practical way. By having to apply the concepts learned in lectures to small practical projects, students not only interiorize concepts more easily, but also acquire a practical, hands-on view of security, which also helps to keep their motivation. The assignment may vary in difficulty and depth, from simple, highly-guided labs to small project-like assignments where students have to solve a more generally specified problem, such as securing a web server or designing a firewall architecture for a given fictitious corporation. Labs can be very motivating and rewarding for students, but it is very difficult to adjust their difficulty. Often small assignments are too mechanic to be a challenge, and complex projects may be frustrating and exceed the scope of the course (e.g. securing a web server would usually require to install the web server, which may be a time-consuming task not directly related to the course).

An alternative to have security labs which are motivational, instructive and affordable for the students is to devise a number of security-related scenarios which reproduce real problems of the discipline. These scenarios, if well designed and implemented, could allow students to face realistic assignments specifically tailored to suit their learning needs and capabilities, as has been successfully proven for other disciplines (Siddiqui et al., 2008). In the following we

discuss the requirements which such scenarios must have and propose an approach to facilitate their implementation.

3. Using Scenarios for Education on Network and System Security

In this section we review the key challenges that arise when designing a security lab based on scenarios, and we analyze the main existing approaches in this field.

3.1. Challenges

When designing a security lab with the goals stated in the previous section, we need to take into account some considerations (Yang et al., 2004):

- *Protection of the campus networks:* As we have stated before, system administrators are usually reluctant to the deployment of security labs due to the potential impact they may have on the security of the campus network. For instance, some assignments related to penetration testing or vulnerability analysis skills may require deploying instances of vulnerable versions of services, which could cause an increase in the number of security incidents or the spread of malware infections. To avoid this, a common practice is to isolate these labs from the rest of the network.

- *Access to Internet:* However, in most cases students may need Internet access to complete their assignments, since looking for information in the Internet is one of the basic learning ways for many disciplines, and particularly for computer security. Specific hosts may be used to this end while keeping the lab machines isolated, but this may impose some unrealistic overheads on the students (e.g. computer switching, data transfer from the Internet-enabled hosts to the isolated ones...).

- *Simulation of corporate-scale networks:* A typical corporate environment will present a wide range of hardware and software elements, such as routers, switches, firewalls or different kinds of servers. If the company is concerned with secure access, we may find additional components like VPN or RADIUS servers (Rigney et al., 2000). It is also usual to find wireless networks in these environments. Deploying and maintaining an infrastructure containing all those elements for education on network security usually exceeds the available resources (both material and human) at most education institutions.

- *Sharing resources for different lab assignments:* Another recurring problem is the difficulty to support different scenarios simultaneously. For instance, we may think about different security-related subjects from different courses, which may have different assumptions and scenarios for their assignments, and they may need to share the lab facilities. Even for the same course and lab, an specific assignment may require the students to experiment with different network configurations (e.g. when studying firewall architectures). Given the usual limitations on equipment and administration staff availability, this is not always possible. Some authors address these challenges by using virtual LANs and removable hard drives in the network hosts (Lee et al., 2011). In this case, the network is in a VLAN different from the campus network, and each subject works with a different set of hard drives. Reconfiguring the VLAN may also allow to emulate different network environments.

- *Resources needed for the assignments:* Students may not have at hand the required equipment to reproduce the proposed scenarios for the different assignments, but it would be desirable that they could reproduce at least some of the environments in their own machines without the need for additional components, or that they are able to access the lab infrastructure remotely from their own networks.

- *Use of state-of-the-art technologies:* A key aspect of information system security is its dynamicity. New systems and technologies are continuously appearing; new vulnerabilities are discovered every day. To provide a security education matching the real needs of the society, it is necessary that our education environments adapt to these technological changes, like new services, network technologies, operating systems or attack techniques. To achieve this, the security lab design proposed must be extensible, in the sense that it must be easy to add new components to the existing network. Extensibility and scalability are key issues in order to be able to provide realistic and state-of-the-art scenarios in the labs.

- *Overhead imposed by the configuration and maintenance of the lab for different assignments:* Configuring and maintaining the security labs to be used in different subjects may require a vast amount of resources. Thorough planning of the installation and deployment procedures for the different lab configurations is required to speed up maintenance tasks. Even systematically automating these processes, maintenance tasks on such labs may be overwhelming.

3.2. Existing Approaches

Most of the existing approaches for security labs present some limitations on several of the aforementioned aspects. Here we will briefly review the main existing proposals, dividing them into three categories: hardware labs, decentralized virtual labs and centralized virtual labs.

3.2.1. Hardware Labs

The way to achieve the maximum level of realism in the design of a security lab is, of course, to use real hardware. Students may use real devices (including real network components) and experience the problems derived from the use of such components. Though, as we have said, this kind of labs provide the more realistic experience, they raise also important concerns. The main drawback is related to cost: all components present in the scenario need to be acquired. There are also costs related to the time needed to install and configure each scenario we want to work with. Finally, we have to take into account portability: students must work on the real scenario and they cannot reproduce the environment easily. The teaching staff also need to invest a significant amount of time in redeploying the scenarios for different courses, subjects or assignments. Therefore, this kind of labs, even providing the closest experience to a real environment, may exceed the available resources of most institutions. An example of this kind of approach is the *Georgia Tech's Hands-On Information Security Lab* (Abler et al., 2006). In this environment, taking advantage of the capabilities of Cisco network equipment, it is even possible to reconfigure the network infrastructure to a certain extent. However, the authors acknowledge that the requirements imposed by their approach make it unfeasible for many medium and small institutions.

3.2.2. Decentralized Virtualization

Many teachers do not see the hardware security labs feasible or suitable for their purposes, due to all the drawbacks discussed in the previous section. Virtualization technologies may help to deploy similar scenarios in a more efficient manner (Romero-Zaldivar et al., 2012). With this model, we can use the existing hardware infrastructure to launch virtual machines which constitute the desired security scenario. These virtual machines can be deployed in different physical machines, and may replicate different network architectures. With this approach we can achieve a higher degree of stability and fault tolerance: since the state of the virtual machines may be saved, students may work more safely with the scenario. If destructive consequences happen (intentionally or accidentally), we can always return to the latest stable state.

There are basically two different models to implement such a lab. On one hand, we can store virtual machine images in a centralized storage system. When a student wants to launch a scenario for a given project or assignment, he will download the appropriate images to his local physical machine and launch these images with the virtualization engine used. A key advantage of this model is the abstraction from the real hardware by means of the virtual machines. Another strength of this approach is that each student may reproduce his own set of virtual machines in an isolated environment. However, this strategy presents some important drawbacks, like the required download time prior to launching the scenario, and the limitation to the maximum number of nodes the scenario may have, which will be given by the computation, memory and storage capabilities of the local physical machine. To address these drawbacks we may devise a different model where, instead of downloading a set of virtual machines to each workstation, a larger, single set of images is distributed among a set of workstations. In this way, if we could run six virtual machines per workstation, for instance, we could simulate a network with 30 hosts using five workstations. The main advantage of this approach is to allow for resource aggregation to simulate larger networks, thus getting closer to the experience of working in a real lab. Regarding network configuration, plain topologies may be deployed in an straightforward manner. For more complex scenarios, we need to resort to techniques allowing link-level virtualization, like Virtual Distributed Ethernet (VDE), which allow host interconnections using virtual switches and routers. Examples of this kind of labs are Purdue ReAssure lab (ReAssure, 2012) and TinkerNet (Winters et al., 2006).

3.2.3. Centralized Virtualization

Other models propose to use a centralized approach to provide a virtual network environment (Romero-Zaldivar et al., 2012). A central server hosts the virtual networks for all the students. Though in theory this centralized server could support either complete virtualization or OS virtualization, for relatively complex scenarios computation limitations will typically force the use OS virtualization. Students can access the central server and create virtual networks up to a moderate scale with low impact on the server. In addition, this also allows the student to access the proposed scenarios remotely, with the increased convenience for time management. In this kind of labs, clusters may be used to increase scalability. The main strength of this approach is the increased simplicity for the management of the lab. However, it relies on a single centralized server, with the associated availability risks, and network connectivity to this server is required in order to be able to work with the scenarios.

4. A Scenario Generation Framework for Security-related Education

As we have seen, most existing approximations present some limitations, specially regarding flexibility, scalability and resource consumption. To address these limitations, in this section we propose NEMESIS (Network EMulator for Education on System and Internet Security), a scenario generation framework for security-related education. First, we establish the design requirements which have guided the development of the framework. Then, we describe the main elements of our proposal. Finally, we briefly outline the most relevant details about the implementation of the framework and the way scenarios are defined.

4.1. Design requirements

When we first undertook the development of a scenario generator to be used for security-related subjects, we thought on one hand to provide a framework with the power and flexibility to assist us in the whole teaching process: lectures, team assignments, evaluations... On the other hand, we wanted it to be able to reproduce (and enhance) all the practical activities we already used in our subjects. Taking this into account, NEMESIS design evolved around the following requirements:

- *Expressiveness:* The framework must allow to design scenarios illustrating all concepts covered in security-related subjects, such as vulnerabilities, threats, risks and security mechanisms.
- *Interactivity:* Students must be able to interact with the scenarios in a way that allows them to acquire and practice the different skills involved in security-related subjects, such as vulnerability analysis, remote access, privilege escalation, intrusion detection, forensics analysis or security audit.
- *Event management:* One of the major limitations of existing frameworks is that the generated scenarios are static or asynchronous, that is, they do not allow for event generation in real time. Many real attacks (and the reactions to them) are built on the execution of a succession of steps throughout a given time interval or on response to certain stimuli. Taking this into account, the framework must be able to simulate event timelines to illustrate these processes.
- *Transparency:* Using the virtual scenario generation tool must not impact negatively the student experience when facing the scenario. The interaction between the student and the scenario, and the results obtained from this interaction, must be identical to what he would obtain working with the real scenarios (except, probably, regarding time measurements).
- *Scalability:* The framework must allow for the generation of complex scenarios, with multiple networks, hosts and services.
- *Security and Accountability:* Apart from robustness and fault tolerance, in order to enable the use of the framework as a continuous student evaluation, the framework must present its own security and audit mechanisms, which allow to track student progress and ensure that he may not alter the scenario or take advantage of it.

- *Openness and extensibility:* Network and system security is an extremely wide field in continuous growth. New vulnerabilities, new attack techniques and new security mechanisms appear every day. For the framework to be useful in the long term, it must support the possibility to add extensions to enable education in more specific environments (e.g. database security) or to address successive advancements of the discipline.

- *Backwards compatibility:* Using this framework must not limit teaching, so it must support at least the functionality to reproduce classical security lab assignments, such as *ARP spoofing*, privilege escalation *wargames* or forensic analysis challenges.

4.2. Key elements

In order to fulfil the requirements discussed above, we have chosen a modular architecture to facilitate expressiveness and flexibility of the framework, and to ensure its extensibility. The key elements we have considered in NEMESIS are the following:

- *Host:* Each one of the machines in the considered network scenario. General features of the deployed machines (e.g. operating system, version) may be selected among the different available templates. One of the basic extension mechanisms for the framework is the development of new machine templates.

- *Network Segment:* encompasses a set of *hosts*, which have connectivity at link-level.

- *Gateway:* A particular type of *host*, which acts as the connection between two or more network segments. The same templates available as *hosts* may be used as *gateways*.

- *Service:* Each *host* may run one or more services, such as *telnet*, *ssh*, or *http*, which may be selected from a library of available services. Creation of new services is another mechanism to extend NEMESIS.

- *Event:* For each *host*, events or sets of events may be programmed. An event is just a sequence of actions (e.g. running a program, launching a service), which are triggered at a given time.

- *Attack:* Among the different actions which may be triggered by events, attacks are specially relevant for the teaching goals of the framework. Attacks may be selected from an attack library, and new attacks may be developed as extensions for the framework.

The different logical elements in a NEMESIS scenario, along with the relationships between them, may be seen graphically in Fig. 1.

4.3. A Distributed Architecture of Virtual Machines

The choice of a hardware and software architecture for NEMESIS has been clearly conditioned by three essential aspects: the diversity of machines and configurations it should be able to handle, the scalability requirements imposed on it, and the need to meet certain security requirements for the framework itself.

It is obvious that we cannot expect to have in our lab each and every type of machine which may be interesting from the security education point of view. A particular type of attack may only be effective for a given version of an operating system, which may not match the one in the lab or in the students' laptops. This suggests the use of virtual machines (Smith & Nair,

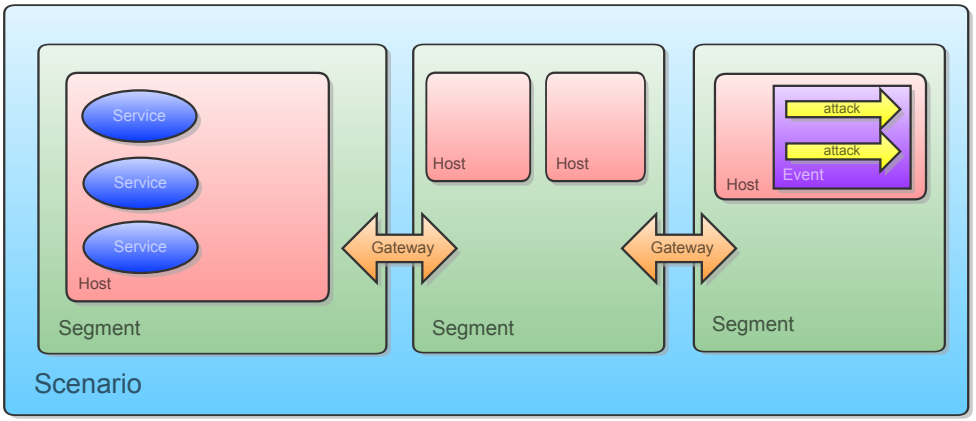

Figure 1: Elements of a NEMESIS scenario

2005a) to implement hosts with the required features regardless of the physical machines we are using. However, the use of virtual machines has some significant drawbacks. First of all, we have performance. If we want to simulate a big network scenario (e.g. 50 machines), we may easily exhaust the computing and memory resources of the host machine, making very difficult, or even unfeasible, to work with the scenario. On the other hand, the fact that the scenario runs using virtual machines in a physical machine which is controlled by the student may raise some issues regarding student evaluation. Of course, this may be addressed by limiting the use of the framework to the lab machines (owned by the teaching institution), but that would limit the possibilities of using NEMESIS as a support tool for e-Learning.

Taking this into account, we propose an architecture based on virtual machines which may be distributed across different physical machines. Thus, each physical machine will host one or more network segments, which in turn will contain one or more virtual machines, supported by a virtual machine manager (VMM) (Smith & Nair, 2005b). Network segments will be interconnected by means of their corresponding virtual *gateways*, so that for the *hosts* the fact that the scenario is distributed across different machines is totally transparent. The resulting architecture is shown in Fig. 2. In the diagram we may observe also the existence of a *management agent* (MA), which is entitled with scenario launching and management. There is also a *deployment agent* (DA) in each physical machine, which starts network segments and virtual machines on request from the MA, and establishes the necessary channels for the communications between the different segments.

With a distributed virtual architecture as the one described, performance problems may be mitigated by distributing computational load among different machines. On the other hand, this architecture allows, for instance, to deploy the virtual machines from which students will act as "attackers" in different host machines from the ones hosting the network segments they have to attack, preventing in this way scenario manipulation.

The implementation of the proposed distributed architecture raises some additional challenges, such as controlling the startup and stop of the virtual machines, the adequate isolation between different network segments and the confidentiality of the communication among physical machines. These challenges have conditioned to a great extent the virtualization solution adopted. Though we studied the possibility of using VMWare (VMWare, 2012) and Xen (Xen,

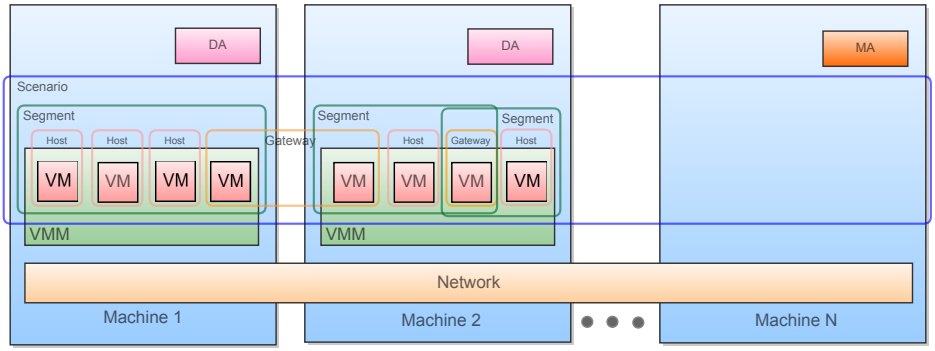

Figure 2: Distributed architecture of virtual machines in NEMESIS

2012), the finally chosen solution was KVM (KVM, 2012), due to its greater flexibility and adaptability to the specific needs of the framework.

4.4. Scenario description, startup and management

Once we have an architecture of virtual machines which may be interconnected in a transparent and secure manner, the next step is to define the way scenarios can be described. Taking into account the flexibility and extensibility requirements, we have chosen to describe the scenarios using XML files. Fig. 3 shows an example of a description file for a basic scenario.

Fully detailing the XML syntax considered for scenario description is beyond the scope of this article. We will use, however, the example file shown to illustrate some relevant details about the framework and its expressive potential:

- The XML scenario file is loaded by the management agent MA, which will parse it and send the required commands to the deployment agents in the physical machines, so that each machine deploys its corresponding part of the scenario. Currently, the deployment agents and management agents are implemented in C++ and Python. C++ is used for its efficiency, while Python is used for its flexibility in parsing tasks.

- Each network segment, defined by the element `segment`, is launched in a given physical machine, identified by the attribute `ip`. If several segments are launched in the same physical machine, a port must be specified as an additional attribute for each segment. This port will be used for communications involving the virtual machines belonging to that segment.

- Hosts may be included within each segment, defining them with the element `host`. For each *host*, we must specify the `template` defining the kind of virtual machine to be launched. The different templates are previously stored in the physical machines. It is also possible to specify some generic configuration attributes, such as the IP address.

- The element `config` may be used to specify more detailed and complex configurations in hosts, gateways and servers, among others. The syntax of the configuration file will depend on the specific template, to allow for a higher degree of customization. The configuration file is parsed by the deployment agent to configure the virtual machine after its launch.
- By default, virtual machines are launched without a graphical interface. The element `display` may be used to enable a graphical interface for a specific host (e.g. so that a student may operate through it) and to specify which physical machine will launch the interface. This allows to export host interfaces to machines not belonging to the scenario, which allows to interact with the scenario without giving access to the machines the user should not have access to, even if they are in the same virtual network segment.
- Network segments are connected by means of *gateways*, using pre-stored templates, in a similar way to hosts. In the example we can see there is a `firewall` template where we can specify packet filtering rules.
- Within each host, we may use the element `service` to define services to be run at virtual machine startup. Among the direct configuration attributes we can specify the port (or ports) for service execution and the user to run the service as. For more complex configurations we can use the element `config`.
- Events (element `event`) may be defined to be run at the host at an specific time (in the example, 5 minutes and 16 seconds after the scenario startup).
- Finally, among the different actions which may be defined for events, in the listing we can show an example of an `attack`, where a *port scan* is launched against a given machine, specifying the port range for the scan.

The sample listing shown illustrates NEMESIS functionality from a technical point of view, and clarifies some relevant details about its implementation. In addition, collection of videos showing the usage of the framework can be found at `http://www.youtube.com/user/telematicauah`. In any case, we must also assess the possibilities of the framework from the point of view of education on network and system security, which we cover in the next section.

5. Using NEMESIS as a support tool for education

In this section we discuss NEMESIS possibilities for teaching security-related subjects. To do this, we present two different examples of what could be typical assignments for system security labs. The first presents a group assignment which can even involve some degree of competition, while the second focuses on the possibilities of the framework to provide automated feedback for students and assessment help for instructors.

5.1. An "attack and defend" methodology

"Attack and defend" security assignments take the form of little *war-games*, where students are divided in two teams (or several groups of two teams), which are assigned either the role of "Black-Hat" (attackers) or "White-Hat" (defenders). The main goal of the offensive teams is to compromise the security of the systems in the scenario, while the defensive teams are in charge of protecting the information assets hosted by those systems, in terms of confidentiality, integrity

and availability (Hill et al., 2001). Along with the inherent realisms of this kind of settings, it benefits from the impact that competitive learning has on the motivation, interest and satisfaction of students (Burguillo, 2010).

The following is an example of an "attack and defend scenario" built using NEMESIS. As stated above, the lab may be seen from two perspectives: as the *attacker* and as the *defender*. The attacker assignment would be to gain access to a set of sensitive data stored in a LDAP server within an enterprise corporative network. The defender assignment would be to analyze the network infrastructure and to deploy the adequate security mechanisms to avoid or counter the attack, or to mitigate the associated losses.

An interesting advantage of using NEMESIS is the possibility to automatize both parts of the scenario, so that a student may participate in the scenario as either an attacker or a defender, alternate the roles or even watch as a spectator while the scenario evolves automatically (e.g. to compare his solution to the one considered by the teaching staff).

5.1.1. Scenario from the attacker's perspective

In order to have a more didactic attack scenario, we chose to launch a model attack in six stages (McClure et al., 2012): reconnaissance, remote access, privilege escalation, attack payload, access maintenance and covering tracks. For the host from which the attacker operates, we have used a virtual machine template based on the Linux distribution *Backtrack* (Backtrack, 2012), specifically designed to perform penetration testing in network and system security. For the attacked hosts we have used templates based on *Damn Vulnerable Linux* (DVL) (Linux, 2012), with the modifications and configuration files needed to adapt them to the scenario needs. The actions considered for the attack in each stage are as follows:

1. *Reconnaissance:* Analysis of the corporate web of the target enterprise. Students playing the role of attackers may perform a manual analysis. To automate the attack, we use a Python script acting as a *web crawler* (Vega-Gorgojo et al., 2010) which downloads the content of the corporate website to the attacker host, and then uses pattern matching filters to detect strings with potentially sensitive information. In this case, the sensitive information is the IP address of another publicly accessible host in the corporate network. A port scan will reveal a SMTP server running in this machine.
2. *Remote access:* The SMTP version running in the server has an input validation vulnerability allowing to execute short shell commands. Students may try to exploit this vulnerability to open a *reverse telnet channel* allowing to issue arbitrary shell orders to the victim machine. For the automated version of the attack, these actions are executed using *Metasploit* (Maynor & Mookhey, 2007), a vulnerability exploitation framework included in *Backtrack*. The victim machine is behind a firewall, so the *telnet* channel must be established through allowed ports.
3. *Privilege scalation:* The SMTP server runs as a non-privileged user (`nobody`). However, there is a executable file which runs some functionality with superuser privileges, and a race condition vulnerability (Dean & Hu, 2004) which allows local privilege escalation towards `root`. Again, automated exploitation is performed using *Metasploit*.
4. *Attack payload:* The LDAP server stores the sensitive information in encrypted text fields. The key used for encryption is the `root` password (there are clues in the LDAP configuration files allowing to deduce this). The encryption key can thus easily be extracted using a password cracking tool over the `/etc/shadow` file. From the available encrypted data

in the LDAP server, students may deduce the encryption algorithm used, or at least reduce the list of algorithms to an acceptable number. In the case of the automated attack, all reasonable encryption mechanisms are tried until an adequately formatted plain text is obtained.

5. *Access maintenance:* In order to be able to access the victim machine again if necessary, the attacker installs a *rootkit* (Hoglund & Butler, 2005) which leaves an open backdoor while hiding its existence. The chosen rootkit for the lab is *SuckIT* (Riley et al., 2009).
6. *Covering tracks:* Finally, before closing the connection to the victim machine, the attacker modifies the log files to erase all potential traces of the attack. In the automated version of the attack, this is done by means of shell scripts.

5.1.2. Scenario from the defender's perspective

From the defender's point of view, the lab assignment may be devised in two different ways. The scenario may be started without launching the attacker machine, and we may allow the defender (be it the student or the automated version of this part of the scenario) to deploy a set of security mechanisms, and then launch the attacker's machine and start the attack (again, either automated or manual). Another possibility is to launch the complete scenario and program the attack for a given time, turning the lab assignment into a countdown race.

The possibilities considered for the scenario in the defender's part are as follows:

- *Vulnerability analysis:* Either manually or using automated tools (e.g. *Nessus*), most of the vulnerabilities in the corporate network may be detected.
- *Firewall configuration:* The defender may try to modify firewall rules to make them more adequate (e.g. blocking potential reverse telnet channels).
- *Intrusion Detection Systems:* The *Snort* IDS (Koziol, 2003), available in the victim machine, may be enabled to detect some of the most usual attack vectors. *Snort* may be even configured to modify the firewall access rules in real time.
- *Gathering of evidence:* The different machines in the corporate network may be configured to store system and audit logs with different sensitivity levels, or we may link the logging sensitivity levels to the events detected by *Snort*. We can also configure the machines to store logs in a dedicated server, or even in write-once media (e.g. CD). This will not thwart the attack or mitigate its consequences, but it allows for forensic analysis, which (apart from its utility in a real investigation) has a great didactic value for the student, which will be able to examine the footprints from his attack (or from the automated attack).
- *Honeypots:* As a particular case of evidence gathering, we can enable *honeypots* (Levine et al., 2004) in the corporate network machines. We may also link the *honeypot* activation to the detection of specific events by *Snort*, making them work as *padded cells* (Bace, 2000).

5.2. Taking advantage of NEMESIS flexibility for assessment and feedback: a Perimeter Security scenario

In this section we will describe how to employ NEMESIS to design a lab related to perimeter security. The choice of this specific environment is due to the special challenges this scenario

implies because of both the deployment of the lab and the design of proper activities for the students to be evaluated. First, due to its very nature, perimeter security requires different network segments to be defined, each of them having their own addressing and interconnected by means of switches and routers. Second, while it could be feasible to emulate this kind of scenarios using other alternatives, there still would be great problems for the students regarding configuration and management. According to our own teaching experience, this burden sometimes keeps the student from focusing on the learning objectives of the subject. Third, to evaluate and validate the results from this kind of environments, special traffic patterns (e.g., for testing DDoS defense capabilities) have to be generated that could interfere in some of the deployed network security tools usually deployed in campus networks. Finally, we would like to enable the student to switch between different choices for perimeter security (e.g., multiple screened subnets, split subnets...). Therefore, we consider that this kind of scenarios could benefit greatly from NEMESIS and, accordingly, NEMESIS features could be shown clearly.

Traditional security courses taught at our university have usually included perimeter security labs. Usually, from a 60-hour course, four hours (two sessions, each of one being two hours long) were devoted to perimeter security labs, starting with firewall security and ending with intrusion detection systems (IDS). Those labs were based on single-box firewall architectures consisting of a screening router working as packet filtering device, commonly using GNU Linux, and a bastion host in the local network. The packet filtering on the screening router had to be set up in such a way that the bastion host should be the only system on the internal network that hosts on the Internet can open connections to (for example, to deliver incoming email). Even then, only certain types of connections were allowed. Any external system trying to access internal systems or services would have to connect to this host. Moreover, other internal hosts were considered in the scenario. The student was required to configure the screening router to allow internal hosts to open connections to hosts on the Internet for certain services and to disallow all connections from external networks to services other than the one the bastion host is offering. In the second part of the lab session, new threats, which could not be easily tackled using a traditional firewall (e.g., password cracking attacks), were introduced and the use of an intrusion detection system (IDS) was suggested. The student was required to create rules for the IDS *Snort* (Koziol, 2003) to deal with this kind of new threats. Either in the firewall or in the IDS scenario, students were required to set up the environment and, after this, an evaluation was performed to check the compliance with the lab requirements, usually by hand. The combined evaluation of both formal correctness and requirement compliance was used to evaluate the student work.

This mechanism, while able to meet the basic requirements for these sessions, has some drawbacks, though. On the one hand, there are limitations regarding the complexity of the scenarios that can be deployed using such an approach, posed mainly by the problems derived from setting up complex network configurations by hand (several networks interconnected by routers/firewalls) either on virtualization environments or with real systems and networks. From a teaching point of view, this establishes a limit in the type of environments we are able to work with, preventing us from working with, for instance, architectures with multiple screened subnets. On the other hand, evaluating student's work is very time consuming because most of the testing has to be done by hand, which results in a significantly high workload for the teaching personnel.

NEMESIS can be used to overcome these limitations. To illustrate the benefits derived from its use, we present a scenario we have been using so far, which models a typical use case scenario: a big corporation where different areas (administration, R&D, Internet servers ...) have different, even separate, networks with different requirements regarding access control and network

security. The complexity of this scenario justifies the use of NEMESIS.

The student is presented this use case with the specific requirements for each of the network areas and is provided with some design alternatives from where he is required to choose one of them according to the requirements and her own understanding of the proposed scenario. These alternatives have been previously created by the course teaching staff. Both the scenario description (as an XML file) for each of the alternatives and the required virtual machines (bastion host, workstations, gateway ...) are provided, thus the student has only to deal with IP address management, which can be performed easily by modifying the XML scenario description. Once this is completed, the student can start the scenario emulation.

Over this scenario, the student is required to design and implement perimeter security mechanisms using, in a first stage, only firewalls and later adding intrusion detection systems. By means of using NEMESIS gateway templates (that have been mentioned before), it is possible to focus only on configuration and design issues and forget about installation aspects. The gateway template is based on GNU Linux and includes both *iptables* (Netfilter, 2012) (for packet filtering management) and *Snort* software packages.

To realize this lab, we have devised two different ways of working with NEMESIS. First, it is possible to implement the security configurations (firewall and IDS rules) by means of a high level description where the students do not have to deal with low-level IDS or firewall configuration. This alternative is the preferred choice for courses where students have little background on OS administration. This high level description allows the definition of either filtering rules (*'allow http protocol on port 80'*) or rules for an IDS describing threats (*'allow less than 5 login attempts'*) without knowing how these rules are actually implemented. This approach makes possible to focus on the design issues of this kind of systems regardless of the implementation aspects. Second, a low-level OS-specific description is also available and can use as much expressivity as the underlying system allows. Both kinds of descriptions could be either included in the XML scenario description file or in another XML file that would be referenced from the former and accordingly loaded when necessary: the student could load and unload different sets of firewall rules just to see how well they perform. In summary, the choice of either high-level or low-level descriptions would depend on the nature of the course to be taught and the students involved.

After the design and implementation stages, it is also possible to define automated validation and evaluation mechanisms. When defining the scenario, validation routines can also be included in order to assess the compliance with the defined requirements. These validation routines, which are also defined in external XML files, are implemented through automated tests that have to be defined in advance. To perform the actual validation, these tests take network parameters from the scenario description so that the teaching personnel do not have to know in advance the detailed network-addressing scheme chosen by the student. The purpose of this kind of routines is twofold: first, a formal validation is conducted over the set of rules defined by the student in order to evaluate whether they are consistent or not. Only after this validation, the requirement-compliance check is performed. This check consists of a set of automated checks, which emulate different kind of network traffic patterns, either legitimate or not. An example of the former would be *'access host 'web server' using http port 80 from Internet'* and one of the latter *'Do SYN Flood over hosts in DMZ network'*. This validation results in a detailed score reflecting the degree of compliance with the scenario security requirements. This scoring will be useful both for the student, as it provides her with a feedback about how its work is improving, and for the teaching staff as it can be used as another metric to evaluate the student work.

6. Case study: teaching System Security with and without NEMESIS

In order to analyze our experience with NEMESIS in teaching, in this section we describe the main features of security related courses relevant to the experience, the experiment's design and the nature of the data collected and the instruments used to evaluate the experience.

6.1. The educational context

Traditionally, Internet Security (IS) is an optional undergraduate course delivered in the Computer Science academic plan of the University of Alcala (Spain). It includes 30 practical lab hours out of 60. The goal of the course is to provide students with a solid foundation on Information Security, Network Security and System Security. Most theoretical teaching hours deal with cryptography, while practical classes are mostly devoted to Network and System Security. Academic results in the last decade revealed that students acquired a good understanding of the cryptography topics of the course, while the skills related to network and system security were found more difficult to learn. Our hypothesis about that was that system security required a more 'hands on' experience, where students were given the opportunity to face some of the real-world problems of the discipline. Taking this into account, in the last year we rebuilt the practical part of the course to make labs more realistic. However, infrastructure and budget constraints hardly limited the extent to which we could make improvements to the lab assignments.

With the transition to the European Space for Higher Education (ESHE) Spanish Universities are performing a great effort to provide students with a more practical and skill-oriented teaching. We saw this as an opportunity to develop a framework to generate security-oriented lab scenarios in a realistic and flexible manner. The framework described in this document will be put into use for the course Information Systems Security, which is a base course in the new academic plans, with 3 practical European credits out of 6. The contents of the course are equivalent to the Internet Security course mentioned above. This provides a unique opportunity for validation of the framework, since both courses are equivalent, which allows to compare results more rigorously.

6.2. The experiment

The experiment took place during the spring term of the 2011/2012 year. During this term, we taught the traditional Internet Security course with the new methodology devised for the new Information System Security (ISS) course within the framework of the ESHE, which included the use of NEMESIS as a learning tool. In this way, students from the ISS version of the course form the experimental group, while students from the previous year IS course form the control group in a static group comparison design (Fraenkel & Wallen, 2000), without creating any unfairness within the students in the same course. To allow for an even better comparison, students in both courses were assessed in the same way, by means of a series of Continuous Assessment Tests (CAT) distributed throughout the semester. The only difference between both courses was the use of the NEMESIS framework. Finally, there were no significant demographic differences between both courses.

6.3. Instruments and data collection

We evaluated the experiments by means of two instruments. On the one hand, we used the academic results for the 2010/2011 course on Internet Security (IS) and the 2011/2012 course adapted to the methodology of the new Information Systems Security course (ISS). To avoid bias, only data about students that had taken the course for the first time was collected. On the other

	IS course		ISS course	
	mean	std. dev.	mean	std. dev.
Cryptography	7.793	1.029	7.325	1.366
System security	5.660	1.104	7.668	1.491
Global	6.726	0.690	7.497	0.963

Table 1: Academic results for IS (2010/2011) and ISS (2011/2012).

hand, we performed two different surveys to measure student's satisfaction with the NEMESIS framework. One of the surveys was conducted with the first-year ISS students, and the other with the transferred students, that is, the ones which had taken the IS course in the past and took the ISS version this year. The survey was composed of two parts: personal data for statistics (age, gender...) and five-score Likert-type scale items. The survey was completed by all students through an online form in the Blackboard LMS (Blackboard, 2012) platform associated to the course.

6.4. Results and discussion

Table 1 shows the academic results in a scale from 0 to 10 (being 10 the highest qualification) for the traditional Internet Security course for the year 2010/2011 (IS) and the new Information Systems Security version of the course for the year 2011/2012 (ISS). The results are divided into two different parts: cryptography and system security. Total scores are also included for the sake of completeness. The results were found to be significantly different with $p < 0.05$. We can see that the results for ISS students are significantly better than the IS ones. In addition, the main differences are observed in the system security part, which backups the hypothesis that it is the difference in the methodology (i.e. the use of NEMESIS) the factor causing the improvement, and not a bias on students' abilities. From the teacher's perspective, using NEMESIS provided a much more flexible and easy way to design and deploy lab scenarios, which allowed to provide more realistic lab assignments to the students. Another proof of the flexibility and ease of use of the platform is that we ask for some master thesis students to design scenarios for security labs, and they came up with high quality scenarios in a fairly reasonable time. We plan to include these scenarios as optional lab assignments in future editions of the course.

From the student's perspective, the subjective perception about the experience was very positive. However, to get a more objective feedback from students, two surveys were conducted. The first survey was oriented to know how ISS students evaluated the system, and the second one was intended to gather the opinion of students who had transferred from IS to ISS in the last year about how the new methodology compared to the old one. Tables 2 and 3 show the average marks for both questionnaires, where every question was graded in a scale from 1 to 5, which ranged from "Strongly Disagree" to "Strongly Agree" respectively. The results have been aggregated by topic due to space limitations, since the original survey had 50 questions (24 questions for transferred students). The results show that students were significantly satisfied with the framework, and that transferred students value NEMESIS as an improvement over the previous methodology.

Topic	Average Score
Flexibility to work (time, location)	4.603
Coverage of course topics	3.919
Help to consolidate concepts and skills	4.693
Lab diversity and interest	4.503
Perceived realism	3.640
Overall evaluation of the lab assignments	3.824

Table 2: Survey results for ISS students (2011/2012). Sample size: 17 students.

Topic	Average Score
Usefulness to complete assignments	4.076
Improved learning	4.470
Less need for additional clarifications	4.485
Overall preference	3.758

Table 3: Survey results for transferred students (2011/2012). Sample size: 10 students.

7. Conclusions and future work

Realistic design of security labs is a challenge for teachers in this kind of subjects. There exist different alternatives to address this challenge, ranging from purely hardware labs to the use of different virtualization techniques. However, most approaches found in the literature have serious limitations, specially regarding flexibility, extensibility and security. Trying to address these limitations, in this article we present NEMESIS, a framework for the generation and emulation of scenarios for teaching network and system security. The platform relies on existing virtualization techniques to facilitate scenario portability, and presents a modular design to allow extensibility. Moreover, the framework has been designed to allow for scenario distribution among multiple physical machines, which enhances scalability. Finally, scenario design relies on templates for hosts, services, and attacks, and uses XML files for scenario definition. This guarantees a certain degree of expressiveness and flexibility.

Though using this framework to design lab assignments has yielded satisfactory results (as shown in the provided use example), there is still plenty of research to be done in this area. We are working on the creation of more templates for different systems, services and attacks and on improving the expressiveness of the scenario description language. Finally, we are interested in creating a community portal for the development of modules for the framework, where other members of the security community may contribute to the growth of NEMESIS.

References

Abler, R., Contis, D., Grizzard, J., & Owen, H. (2006). Georgia tech information security center hands-on network security laboratory. *Education, IEEE Transactions on*, 49(1), 82 – 87.

Bace, R. G. (2000). *Intrusion Detection*. Macmillan Technical Publishing.
Backtrack (2012). Backtrack linux official website. http://www.backtrack-linux.org/, accessed on July 2012.
Bishop, M. (1993). Teaching computer security. In *IFIP Transactions A (Computer Science and Technology)* (pp. 65–74). Netherlands: IFIP. IFIP TC11 Ninth International Conference on Information Security, IFIP/Sec'93. Computer Security, ,, Toronto, Ont., Canada.
Blackboard (2012). Blackboard lms official website. http://www.blackboard.com/, accessed on July 2012.
Burguillo, J. C. (2010). Using game theory and competition-based learning to stimulate student motivation and performance. *Computers & Education*, 55(2), 566–575.
Cardenas, A. A., Roosta, T., & Sastry, S. (2009). Rethinking security properties, threat models, and the design space in sensor networks: A case study in SCADA systems. *Ad Hoc Networks*, 7(8), 1434 – 1447. Privacy and Security in Wireless Sensor and Ad Hoc Networks.
Dean, D. & Hu, A. J. (2004). Fixing races for fun and profit: how to use access(2). In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium* (pp. 14–14). Berkeley, CA, USA: USENIX Association.
Fraenkel, J. R. & Wallen, N. E. (2000). *How to design and evaluate research in education*. McGraw-Hill.
Higgins, J. (1989). Information security as a topic in undergraduate education of computer scientists. In *Information Security as a Topic in Undergraduate Education of Computer Scientists* (pp. 553–557).
Hill, J. M. D., Carver, Jr., C. A., Humphries, J. W., & Pooch, U. W. (2001). Using an isolated network laboratory to teach advanced networks and security. *SIGCSE Bull.*, 33(1), 36–40.
Hoglund, G. & Butler, J. (2005). *Rootkits: Subverting the Windows Kernel*. Addison-Wesley Professional.
Khambari, M. N. M., Fairuz Iskandar Othman, M., Radzi Motsidi, M., & Faizal Abdollah, M. (2009). A novel approach on teaching network security for ict courses. In *Engineering Education (ICEED), 2009 International Conference on* (pp. 66 –71).
Koziol, J. (2003). *Intrusion Detection with Snort*. Indianapolis, IN, USA: Sams, 1 edition.
KVM (2012). Kvm official website. http://www.linux-kvm.org/, accessed on March 2011.
Lee, C., Uluagac, A., Fairbanks, K., & Copeland, J. (2011). The design of netseclab: A small competition-based network security lab. *Education, IEEE Transactions on*, 54(1), 149 –155.
Levine, J., Grizzard, J., & Owen, H. (2004). Using honeynets to protect large enterprise networks. *Security Privacy, IEEE*, 2(6), 73 – 75.
Linux, D. V. (2012). Damn vulnerable linux official website. http://www.damnvulnerablelinux.org/, accessed on July 2012.
Maynor, D. & Mookhey, K. K. (2007). *Metasploit toolkit for penetration testing, exploit development, and vulnerability research*. Syngress.
McClure, S., Scambray, J., & Kurtz, G. (2012). *Hacking exposed 7: network security secrets and solutions*. McGraw Hill Professional.
Netfilter (2012). The netfilter.org project. http://netfilter.org/ , accessed on March 2012.
ReAssure (2012). Reassure official website. http://projects.cerias.purdue.edu/reassure/, accessed on March 2012.
Rigney, C., Willens, S., Rubens, A., & Simpson, W. (2000). Remote Authentication Dial In User Service (RADIUS). RFC 2865 (Draft Standard). Updated by RFCs 2868, 3575, 5080.
Riley, R., Jiang, X., & Xu, D. (2009). Multi-aspect profiling of kernel rootkit behavior. In *Proceedings of the 4th ACM European conference on Computer systems*, EuroSys '09 (pp. 47–60). New York, NY, USA: ACM.
Romero-Zaldivar, V.-A., Pardo, A., Burgos, D., & Kloos, C. D. (2012). Monitoring student progress using virtual appliances: A case study. *Computers & Education*, 58(4), 1058 – 1067.
Sharma, S. K. & Sefchek, J. (2007). Teaching information systems security courses: A hands-on approach. *Computers & Security*, 26(4), 290 – 299.
Shaw, R., Chen, C. C., Harris, A. L., & Huang, H.-J. (2009). The impact of information richness on information security awareness training effectiveness. *Computers & Education*, 52(1), 92 – 100.
Siddiqui, A., Khan, M., & Akhtar, S. (2008). Supply chain simulator: A scenario-based educational tool to enhance student learning. *Computers & Education*, 51(1), 252 – 261.
Smith, J. & Nair, R. (2005a). The architecture of virtual machines. *Computer*, 38(5), 32 – 38.
Smith, J. & Nair, R. (2005b). *Virtual Machines: Versatile Platforms for Systems and Processes*. Morgan Kaufmann.
Vega-Gorgojo, G., Bote-Lorenzo, M. L., Asensio-Pérez, J. I., Gómez-Sánchez, E., Dimitriadis, Y. A., & Jorrín-Abellán, I. M. (2010). Semantic search of tools for collaborative learning with the ontoolsearch system. *Computers & Education*, 54(4), 835 – 848.
VMWare (2012). Vmware official website. http://www.vmware.com/, accessed on March 2011.
Winters, T., Ausanka-Crues, R., Kegel, M., Shimshock, E., Turner, D., & Erlinger, M. (2006). Tinkernet: a low-cost and ready-to-deploy networking laboratory platform. In *ACE '06: Proceedings of the 8th Austalian conference on Computing education* (pp. 253–259). Darlinghurst, Australia, Australia: Australian Computer Society, Inc.
Xen (2012). Xen official website. http://www.xen.org/, accessed on March 2011.
Yang, T. A., Yue, K.-B., Liaw, M., Collins, G., Venkatraman, J. T., Achar, S., Sadasivam, K., & Chen, P. (2004). Design

of a distributed computer security lab. *J. Comput. Small Coll.*, 20(1), 332–346.
Yurcik, W. & Doss, D. (2001). Different approaches in the teaching of information systems security. In *Security," Proceedings of the Information Systems Education Conference* (pp. 32–33).

```
<segment label=interna, host=172.29.16.21>
   <host ip=10.0.10.5, template=windows/>
   <host ip=10.0.10.6, template=windows/>
   <host ip=10.0.10.7, template=windows>
      <config file=sysadmin.xml/>
      <display host=172.29.16.22/>
   </host>
   <gateway template=firewall ipin=10.0.10.1
                ipout=10.0.11.2 linkto=dmz>
     <ipchains rule="allow tcp 80 outbound"/>
     <ipchains rule="allow tcp 25 outbound"/>
     <ipchains rule="allow tcp 22 outbound"/>
   </gateway>
</segment>
<segment label=dmz, host=172.29.16.20>
   <host ip=10.0.11.5, template=dvl>
      <service name=www, port=80, user=root/>
      <service name=sshd/>
   </host>
   <host ip=10.0.11.6, template=dvl>
      <service name=smtp/>
      <service name=sshd/>
   </host>
   <gateway template=firewall ipin=10.0.11.1
          ipout=213.18.21.77 linkto=interna>
      <ipchains rule="allow tcp 25 inbound">
      <ipchains rule="allow tcp 80 inbound">
   </gateway>
</segment>
<segment label=internet, host=172.29.16.19>
   <host ip=213.18.21.7, template=windows/>
   <host ip=213.18.21.70, template=dvl>
      <event at=5m16s>
         <attack template=portscan>
            <param name="ip"
                   value="213.18.21.77"/>
    <param name="range"
                   value="1..1024"/>
         </attack>
      </event>
   </host>
   <gateway template=plain linkto=dmz/>
</segment>
```

Figure 3: Example of an XML file for scenario description.